UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/631,308	07/31/2003	Gerard Chauvel	TI-35433	1887

23494 7590 02/07/2008

TEXAS INSTRUMENTS INCORPORATED
P O BOX 655474, M/S 3999
DALLAS, TX 75265

EXAMINER
PETRANEK, JACOB ANDREW

ART UNIT	PAPER NUMBER
2183	

NOTIFICATION DATE	DELIVERY MODE
02/07/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

uspto@ti.com
uspto@dlemail.itg.ti.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Application Number: 10/631,308
Filing Date: July 31, 2003
Appellant(s): CHAUVEL ET AL.

Utpal D. Shah, Reg. No. 60,047
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 11/14/2007 appealing from the Office action mailed 8/14/2007.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Feierbach et al. (U.S. 6,088,786), Batten et al. (U.S. 6,256,725), Patel et al. (6,826;749), Gee et al. (U.S. 6,374,286), Hennessy et al. ("Computer Architecture: A Quantitative

Approach"), Hendler et al. (U.S. 6,473,777), and Brassac et al. (U.S. 6,928,539) are relied upon as evidence.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Maintained Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 8-11 are rejected under 35 U.S.C. §102(b) as being anticipated by Feierbach et al. (U.S. 6,088,786).

3. As per claim 8:

Feierbach disclosed a method of processing instructions in a processor, comprising:

Fetch logic in a core of the processor receiving instructions from a first instruction set which comprises stack-based instructions (Feierbach: Figure 2 elements 104 and 227, column 5 lines 62-67 continued to column 6 lines 1-7 and column 7 lines 7-18)(Feierbach is silent as to where the PC is maintained, which is read as fetch logic. Official notice is given that the PC for fetching instructions from the instruction cache outside the core could be contained within the core of the processor. Element 227 receives the fetched instructions. The extended instructions are the second instruction set that provides register-based operations on element 204. The regular stack

instructions are the first instruction set that provides stack-based operations on element 202.);

Said fetch logic receiving instructions from a second instruction set which comprises memory-based and register-based instructions (Feierbach: Figure 2 elements 104 and 227, column 5 lines 62-67 continued to column 6 lines 1-7 and column 7 lines 7-18)(Feierbach is silent as to where the PC is maintained, which is read as fetch logic. Official notice is given that the PC for fetching instructions from the instruction cache outside the core could be contained within the core of the processor. Element 227 receives the fetched instructions. The extended instructions are the second instruction set that provides register-based operations on element 204. The regular stack instructions are the first instruction set that provides stack-based operations on element 202.); and

Executing said received instructions from the first and second instruction sets (Feierbach: Figure 2 elements 202 and 204, column 7 lines 27-37)(The execution units execute both the register-based and stack-based operations.).

4. As per claim 9:

Feierbach disclosed the method of claim 8 further comprising forming a sequence of instructions from both of said first and second instruction sets (Feierbach: Column 5 lines 62-67 continued to column 6 lines 1-7)(Instructions that are to be executed can be from both ISA's.).

5. As per claim 10:

Feierbach disclosed the method of claim 8 further comprising an instruction from said second instruction set that targets a stack included in said core, said stack having a top, and storing a value at the top of the stack in a register in the processor (Feierbach: Column 7 lines 50-65)(Official notice is given that stack operations push and pop at the top of the stack.).

6. As per claim 11:

Feierbach disclosed the method of claim 10 further comprising updating an address stored in another register that points to the top of the stack (Feierbach: Column 7 lines 50-65)(Official notice is given that a register stores the pointer to the top of the stack that is updated upon pop and push operations.).

Maintained Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. §103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-2, 4, and 6-7 are rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Batten et al. (U.S. 6,256,725).

9. As per claim 1:

Feierbach disclosed a processor, comprising:

A core (Feierbach: Figure 2 element 104, column 5 lines 40-50);

A multi-entry stack contained within the core and usable in at least a stack-based instruction set and comprising a plurality of entries, all of said entries of said multi-entry stack correspond to a subset of entries at the top of a main stack implemented in memory outside of said core (Feierbach: Figure 2 elements 210 and 224, column 7 lines 8-18 and column 8 lines 2-20)(The stack can pop and push data to the data cache. Element 224 corresponds to memory outside of the core that is fetched for the stack inside the core. The core stack is the top of the stack cache, which is external.);

Logic contained in said core and coupled to said stack, the logic manages the stack (Feierbach: Figure 2 element 202, column 7 lines 50-65); and

A plurality of registers contained in said core and coupled to the logic and addressable through a second instruction set that provides register-based and memory-based operations and (Feierbach: Figure 2 element 208, column 6 lines 1-7 and 53-65)(The extended instructions are the second instruction set that provides register-based operations on element 204. The regular stack instructions are the first instruction set that provides stack based operations on element 202); and

Wherein said core executes instructions from both said stack-based instruction set and said second instruction set (Feierbach: Figure 2 elements 202 and 204, column 6 lines 1-7)(The extended instructions are the second instruction set that provides register-based operations on element 204. The regular stack instructions are the first instruction set that provides stack based operations on element 202.); and

An instruction fetch logic contained in said core, said instruction fetch logic receives at least stack-based instructions from the stack-based instruction set

(Feierbach: Figure 2 elements 104 and 227, column 5 lines 62-67 continued to column 6 lines 1-7 and column 7 lines 7-18)(Feierbach is silent as to where the PC is maintained, which is read as fetch logic. Official notice is given that the PC for fetching instructions from the instruction cache outside the core could be contained within the core of the processor. Element 227 receives the fetched instructions. The regular stack instructions are the first instruction set that provides stack-based operations on element 202.);

Feierbach failed to teach wherein said logic executes instructions from both said stack-based instruction set and said second instruction set.

However, Batten disclosed wherein said logic executes instructions from both said stack-based instruction set and said second instruction set (Batten: Figure 4 element 48, column 5 lines 16-43)(Batten disclosed sharing the execution units to execute both stack-based and register-based instructions. In addition, according to "In re Japikse" (181 F.2d 1019, 86 USPQ 70 (CCPA 1950)), shifting the location of parts doesn't give patentability over prior art.).

The advantage of a shared datapath is that it results in improved performance and a significant reduction in static code size compared to normal processors (Batten: Column 3 lines 22-34). One of ordinary skill in the art would have been motivated by these advantages to combine the datapaths of Feierbach into a single shared datapath. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a shared datapath in Feierbach for the advantages of improved performance and reduction in code size.

10. As per claim 2:

Feierbach and Batten disclosed the processor of claim 1 wherein the stack has a top and the stack is accessible within the second instruction set through at least one of the registers in which a value is stored that is present at the top of the stack (Feierbach: Figure 2 element 208, column 8 lines 33-50)(The stack is accessible through the copy unit that transfers the stack to the register file, where then the register-based instruction have access to the stack values.).

11. As per claim 4:

Feierbach and Batten disclosed the processor of claim 1 wherein the stack-based instruction set accesses operands from the stack and places results from operations on the stack and, as a result of accessing operands the stack and placing results on the stack, at least some of the registers are updated (Feierbach: Column 7 lines 50-65)(Official notice is given that when operands are popped and pushed to the stack, a top of the stack pointer is accordingly updated.).

12. As per claim 6:

Feierbach and Batten disclosed the processor of claim 1 further comprising a pair of parallel address generation units coupled to said logic which are used to compute memory source and destination addresses and wherein a register includes the top of the multi-entry stack, thereby permitting a block of data to be moved between a memory area and the stack by execution of a single instruction with a repeat loop (Feierbach: Column 8 lines 2-20)(It's obvious to one of ordinary skill in the art that the memory operation moving the stack to external memory could be implemented with a repeating

store/load instruction.).

13. As per claim 7:

Feierbach and Batten disclosed the processor of claim 1 wherein the second instruction set comprises an instruction that retrieves operands from memory, performs a computation on the operands, and places the result on the stack (Feierbach: Figure 2 element 208, column 8 lines 33-50)(The register-base operations retrieve operands from the register file, perform a function on the operands, and when placing results in the register that's the top of the stack, has that functionality when the register file is transferred back to the stack by the copy unit.).

14. Claim 3 is rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Batten et al. (U.S. 6,256,725), further in view of Patel et al. (U.S. 6,826,749).

15. As per claim 3:

Feierbach and Batten disclosed the processor of claim 1.

Feierbach and Batten failed to teach wherein the stack has a top that is addressable by a memory mapped address, and the memory mapped address is stored in a register which is accessed by the second instruction set.

However, Patel disclosed wherein the stack has a top that is addressable by a memory mapped address, and the memory mapped address is stored in a register which is accessed by the second instruction set (Patel: Figure 3 element 44, column 5

lines 14-26; Figure 5, column 7 lines 31-54)(The register stores a pointer that points to the top of the stack. Thus having the same functionality.).

Feierbach disclosed a stack and disclosed instructions that pop and push values to the stack, but is silent on the details of addressing the stack properly to ensure the top value is addressed. One of ordinary skill in the art would have been motivated by Feierbach failing to disclose how the top of the stack is addressed to find Patel that uses a register to address the top of the stack. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a stack pointer register for the advantage of being able to properly address the top of the stack.

16. Claim 5 is rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Batten et al. (U.S. 6,256,725), further in view of Gee et al. (U.S. 6,374,286).

17. As per claim 5:

Feierbach and Batten disclosed the processor of claim 1.

Feierbach and Batten failed to teach further comprising a first program counter usable in the execution of the stack-based instruction set and a second program counter usable in the execution of a micro-sequence that comprises instructions from both the stack-based and second instruction sets.

However, Gee disclosed further comprising a first program counter usable in the execution of the stack-based instruction set (Gee: Figure 2 element 250, column 11 lines 1-8)(The PC register when combined with Feierbach is used for the standard stack

instructions) and a second program counter usable in the execution of a micro-sequence that comprises instructions from both the stack-based and second instruction sets (Gee: Figure 2 element 226, column 9 lines 18-26)(Feierbach: Column 6 lines 1-7)(The micro-program register is used for the expanded instructions of Feierbach. It's obvious to one ordinary skill in the art that the expanded instructions are more complex and could be broken up into multiple simpler instructions that the microPC could be used to control. It's also obvious to one of ordinary skill in the art that Feierbach states that the expanded instructions are primarily register-based, but also could be stack-based instructions as well.).

The advantage of using a second program counter for the expanded instructions is that they could be complex instructions that could be broken into multiple simpler instructions to execute on the processor. A second program counter could be used to fetch the simpler instructions from a complex instruction to track the progress of the overall complex instruction. One of ordinary skill in the art would have been motivated by this advantage to include a second program counter to implement the complex instructions. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a second program counter for the complex operations of the expanded instructions of Feierbach for the advantage of keeping track of the progress of the instruction.

18. Claims 12-15, 18-24, and 27 are rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Patel et al. (U.S.

6,826,749), in view of Batten et al. (U.S. 6,256,725) in view of Hennessy et al. ("Computer Architecture: A Quantitative Approach").

19. As per claim 12:

A core (Feierbach: Figure 2 element 104, column 5 lines 40-50);

A multi-entry stack contained in said core and having a top and usable in at least a stack-based instruction set (Feierbach: Figure 2 element 210, column 7 lines 50-65)(Official notice is given that the stack contains a top that is accessed by stack operations.);

Logic contained in said core and coupled to said stack, the logic manages the stack (Feierbach: Figure 2 element 202, column 7 lines 50-65);

Memory coupled to said logic and located outside said core (Feierbach: Figure 1 elements 110 and 108); and

A plurality of registers contained in said core and coupled to the logic and addressable through a second instruction set that provides register-based and memory-based operations (Feierbach: Figure 2 element 208, column 6 lines 1-7 and 53-65)(The extended instructions are the second instruction set that provides register-based operations on element 204. The regular stack instructions are the first instruction set that provides stack-based operations on element 202);

Wherein said multi-entry stack comprises a plurality of entries, all of said entries of said multi-entry stack correspond to a subset of entries of a main stack implemented in said memory (Feierbach: Figure 2 elements 210 and 224, column 7 lines 8-18 and column 8 lines 2-20)(The stack can pop and push data to the data cache. Element 224

corresponds to memory outside of the core that is fetched for the stack inside the core. The core stack is the top of the stack cache, which is external.).

Feierbach failed to teach wherein a first register includes an address through which the top of the stack is accessed and a second register in which a value at the top of the stack is stored; wherein said logic executes instructions from both said stack-based instruction set and said second instruction set; and wherein at least one of the registers are used to calculate addresses in parallel, said addresses being calculated in accordance with any of a plurality of addressing modes specified by the second instruction set.

However, Patel disclosed wherein a first register includes an address through which the top of the stack is accessed and a second register in which a value at the top of the stack is stored (Patel: Figure 3 element 44, column 5 lines 14-26; Figure 5, column 7 lines 30-54)(Optop register is a pointer that points to the top of the stack. The second register is part of the stack that stores the top value.)

Feierbach disclosed a stack and disclosed instructions that pop and push values to the stack, but is silent on the details of addressing the stack properly to ensure the top value is addressed. One of ordinary skill in the art would have been motivated by Feierbach failing to disclose how the top of the stack is addressed to find Patel that uses a register to address the top of the stack. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a stack pointer register for the advantage of being able to properly address the top of the stack.

Feierbach and Patel failed to teach wherein said logic executes instructions from both said stack-based instruction set and said second instruction set and wherein at least one of the registers are used to calculate addresses in parallel, said addresses being calculated in accordance with any of a plurality of addressing modes specified by the second instruction set.

However, Batten disclosed wherein said logic executes instructions from both said stack-based instruction set and said second instruction set (Batten: Figure 4 element 48, column 5 lines 16-43)(Batten disclosed sharing the execution units to execute both stack-based and register-based instructions. In addition, according to "In re Japikse" (181 F.2d 1019, 86 USPQ 70 (CCPA 1950)), shifting the location of parts doesn't give patentability over prior art.).

The advantage of a shared datapath is that it results in improved performance and a significant reduction in static code size compared to normal processors (Batten: Column 3 lines 22-34). One of ordinary skill in the art would have been motivated by these advantages to combine the datapaths of Feierbach into a single shared datapath. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a shared datapath in Feierbach for the advantages of improved performance and reduction in code size.

Feierbach, Patel, and Batten failed to teach wherein at least one of the registers are used to calculate addresses in parallel, said addresses being calculated in accordance with any of a plurality of addressing modes specified by the second instruction set.

However, Hennessy disclosed wherein at least one of the registers are used to calculate addresses in parallel, said addresses being calculated in accordance with any of a plurality of addressing modes specified by the second instruction set (Hennessy: Figures 2.6 and 2.7, section 2.3)(Register addressing mode and register indirect modes for load and stores use a register for calculating the effective address.).

Patel disclosed data memory accesses to the data cache through load and store instruction, but left out the details of the addressing modes used to calculate the memory address needed to access the data or storing the data within the data cache (Patel: Figure 3 element 58, column 5 lines 60-67 continued to column 6 lines 1-3). One of ordinary skill in the art would have been motivated to look for different types of addressing modes that could be used to load and store data from memory. Hennessy disclosed many different types of addressing modes for data memory instructions. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to add the addressing modes of Hennessy to Patel's load and store instructions.

20. As per claim 13:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 12 wherein the stack-based instruction set accesses operands from the multi-entry stack and places results from operations on the multi-entry stack and, as a result of accessing operands from the multi-entry stack and placing results on the multi-entry stack thereby causing the address in the first register to be changed (Patel: Figure 3 element 44 and 50, column 6 lines 43-56; Figure 5, column 7 lines 30-54)(The pointer register and

register holding the top value of the stack are updated each time the stack changed. Thus having the same functionality.).

21. As per claim 14:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 13 wherein the address in the first register is incremented or decremented depending on whether the register is used as a source or a destination, respectively, for an operation (Patel: Figure 3 elements 44 and 50, column 6 lines 43-56; Figure 5, column 7 lines 30-54)(The pointer is incremented or decremented depending on if the top value is popped off the stack or if a value is pushed on the stack. Thus having the same functionality.).

22. As per claim 15:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 12 wherein the stack-based instruction set comprises Java Bytecodes (Patel: Column 4 lines 33-46).

23. As per claim 18:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 12 wherein at least one of the registers includes an offset usable in the calculation of addresses (Hennessy: Figures 2.6 and 2.7, section 2.3)(Register indirect or indexed addressing modes use registers as pointers to the effective address. Displacement addressing mode uses an immediate value to add to a register value for the effective address.).

24. As per claim 19:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 12 wherein the second instruction set comprises an instruction that moves data from a register or memory to a register, and consequently to the multi-entry stack (Patel: Figure 3 element 58, column 5 lines 60-67 continued to column 6 lines 1-3)(Load/store instructions move data to/from the data cache. Thus having the same functionality.).

25. As per claim 20:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 19.

Wherein the instruction that moves data includes a plurality of bits of that encode one of a plurality of addressing modes (Hennessy: Figures 2.6 and 2.7, section 2.3).

26. As per claim 21:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 20 wherein the addressing modes include a mode in which the instruction that moves data includes an immediate value and a reference to a register containing a base address, wherein the immediate value and the base address are added together to generate a source memory address for the move instruction (Hennessy: Figures 2.6 and 2.7, section 2.3)(Displacement addressing mode uses an immediate value that's added to the value contained within a register to obtain the effective address.).

27. As per claim 22:

Claim 22 essentially recites the same limitations of claim 21. Therefore, claim 22 is rejected for the same reasons as claim 21.

28. As per claim 23:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 20

wherein the addressing modes include a mode in which the instruction that moves data includes references to two registers in which memory addresses are stored, one register being a predetermined index register, the memory addresses from the two registers are added together to calculate a source memory address used to complete the move instruction, and the address in the predetermined index register is incremented (Hennessy: Figure 14, column 11 lines 40-67 continued to column 12 lines 1-26)(Autoincrement adds two register values and increments the index register.).

29. As per claim 24:

Claim 24 essentially recites the same limitations of claim 23. Therefore, claim 24 is rejected for the same reasons as claim 23.

30. As per claim 27:

Claim 27 essentially recites the same limitations of claim 6. Therefore, claim 27 is rejected for the same reasons as claim 6.

31. Claims 16 and 26 are rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Patel et al. (U.S. 6,826,749), in view of Batten et al. (U.S. 6,256,725) in view of Hennessy et al. ("Computer Architecture: A Quantitative Approach"), further in view of Gee et al. (U.S. 6,374,286).

32. As per claim 16:

Claim 16 essentially recites the same limitations of claim 5. Therefore, claim 16 is rejected for the same reasons as claim 5.

33. As per claim 26:

Claim 26 essentially recites the same limitations of claim 5. Therefore, claim 26 is rejected for the same reasons as claim 5.

34. Claim 25 is rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Patel et al. (U.S. 6,826,749), in view of Batten et al. (U.S. 6,256,725) in view of Hennessy et al. ("Computer Architecture: A Quantitative Approach"), further in view of Hendler et al. (U.S. 6,473,777) and Brassac et al. (U.S. 6,928,539).

35. As per claim 25:

Feierbach, Patel, Batten, and Hennessy disclosed the processor of claim 12.

Feierbach, Patel, Batten, and Hennessy failed to teach wherein the processor is configured to be coupled to a separate processor on which an operating system is executed.

However, Hendler disclosed wherein the processor is configured to be coupled to a separate processor (Hendler: Figure 2 element 102, column 2 lines 32-46)(Element 102 executes various overhead activities associated with java instructions. Thus having the same functionality.).

The advantage of having another processor to run processes is that it can run overhead tasks, such as compilation and garbage collection, which needs to be done to execute Java instructions (Hendler: Column 1 lines 44-64). The processor running Java instruction will make gains in performance by having another processor handler these overhead tasks (Hendler: Column 1 lines 44-64). The performance increase of a java

stack-based processor would have motivated one of ordinary skill in the art to use another processor for overhead tasks. Thus, it would have been obvious to one of ordinary skill in the art to add another processor to Patel's system that deals with executing overhead tasks for the advantage of increased performance in the java processor.

Feierbach, Patel, Batten, Hennessy, and Hendler failed to teach a separate processor on which an operating system is executed.

However, Brassac disclosed a separate processor on which an operating system is executed (Brassac: Figure 2 element 8, column 1 lines 58-67 continued to column 2 lines 1-7)(The separate processor runs operating system tasks for other processors. Thus having the same functionality.).

The advantage of having another processor to run processes is that it can run overhead tasks, such as compilation and garbage collection, which needs to be done to execute Java instructions (Hendler: Column 1 lines 44-64). Executing instructions for the operating system is another such task that can be considered an overhead task. Running the operating system on the separate processor would have increased the performance of the java processor. One of ordinary skill in the art would have been motivated by this increased performance of the java processor to have the separate processor run OS instructions. Thus, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a separate processor running OS instructions for the added performance to the java processor.

36. Claims 30-35, 37, and 41 are rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Hendler et al. (U.S. 6,473,777).

37. As per claim 30:

Feierbach disclosed a system, comprising:

A co-processor having a core comprises a hardware stack, fetch logic, and registers, said fetch logic receiving stack-based instructions from a first instruction set (Feierbach: Figure 1 element 104, column 5 lines 62-67 continued to column 6 lines 1-7; Figure 2 elements 208, 210, and 227, column 6 lines 53-65)(Feierbach is silent as to where the PC is maintained, which is read as fetch logic. Official notice is given that the PC for fetching instructions from the instruction cache outside the core could be contained within the core of the processor. Element 227 receives the fetched instructions. The fetch logic receives stack based regular instructions and register-based extended instructions.), and the core of the co-processor is configured to execute stack-based instructions from a first instruction set and instructions from a second instruction set that provides memory-based and register-based operations (Feierbach: Figure 1 element 104, column 5 lines 62-67 continued to column 6 lines 1-7)(The fetch logic receives stack based regular instructions and register-based extended instructions.),

Wherein said hardware stack comprises a subset of entries at a top of a memory-based stack implemented in memory outside of said core (Feierbach: Figure 2 elements 210 and 224, column 7 lines 8-18 and column 8 lines 2-20)(The stack can pop and push

data to the data cache. Element 224 corresponds to memory outside of the core that is fetched for the stack inside the core. The core stack is the top of the stack cache, which is external.).

Feierbach failed to teach a main processor unit coupled to the co-processor.

However, Hendler disclosed a main processor unit (Hendler: Figure 2 element 102, column 2 lines 32-46)(Element 102 executes various overhead activities associated with java instructions. Thus having the same functionality.).

The advantage of having another processor to run processes is that it can run overhead tasks, such as compilation and garbage collection, which needs to be done to execute Java instructions (Hendler: Column 1 lines 44-64). The processor running Java instruction will make gains in performance by having another processor handler these overhead tasks (Hendler: Column 1 lines 44-64). The performance increase of a java stack-based processor would have motivated one of ordinary skill in the art to use another processor for overhead tasks. Thus, it would have been obvious to one of ordinary skill in the art to add another processor to Patel's system that deals with executing overhead tasks for the advantage of increased performance in the java processor.

38. As per claim 31:

Feierbach and Hendler disclosed the system of claim 30 wherein stack-based instructions comprise Java bytecodes (Feierbach: Column 5 lines 9-29)

39. As per claim 32:

Feierbach and Hendler disclosed the system of claim 31 further including a

compiler coupled to said co-processor, said compiler receives Java bytecodes and replaces at least one group of bytecodes by a sequence of instructions from the second instruction set and provides said sequence to the co-processor for execution (Feierbach: Column 5 lines 9-29).

40. As per claim 33:

Feierbach and Hendler disclosed the system of claim 32 wherein the sequence also includes stack-based instructions from the first instruction set (Feierbach: Column 5 lines 9-29).

41. As per claim 34:

Feierbach and Hendler disclosed the system of claim 30 wherein the system comprises a cellular telephone (Feierbach: Column 5 lines 40-50.).

42. As per claim 35:

Claim 35 essentially recites the same limitations of claim 2. Therefore, claim 35 is rejected for the same reasons as claim 2.

43. As per claim 37:

Claim 37 essentially recites the same limitations of claim 4. Therefore, claim 37 is rejected for the same reasons as claim 4.

44. As per claim 41:

Claim 41 essentially recites the same limitations of claim 6. Therefore, claim 41 is rejected for the same reasons as claim 6.

45. Claim 36 is rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Hendler et al. (U.S. 6,473,777), further in view of Patel et al. (U.S. 6,826,749).

46. As per claim 36:

Claim 36 essentially recites the same limitations of claim 3. Therefore, claim 36 is rejected for the same reasons as claim 3.

47. Claims 38-40 are rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Hendler et al. (U.S. 6,473,777), further in view of Gee et al. (U.S. 6,374,286).

48. As per claim 38:

Claim 38 essentially recites the same limitations of claim 5. Therefore, claim 38 is rejected for the same reasons as claim 5.

49. As per claim 39:

Claim 39 essentially recites the same limitations of claim 5. Therefore, claim 39 is rejected for the same reasons as claim 5.

50. As per claim 40:

Claim 40 essentially recites the same limitations of claim 5. Therefore, claim 40 is rejected for the same reasons as claim 5.

(10) Response to Argument

51. Regarding claims 1-2, 4, and 6-7 rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Batten et al. (U.S. 6,256,725):

A.) Applicant argues in section A "Feierbach failed to teach a multi-entry stack contained within the core and usable in at least a stack-based instruction set and comprising a plurality of entries and a plurality of registers contained in said core and coupled to the logic and addressable through a second instruction set that provides register-based and memory-based operations. Feierbach is silent to a second instruction set that provides register-based and memory-based operations ... In accordance with Feierbach the extended instructions are one of two "types" of instructions that are part of one set of stack based operations."

The examiner disagrees for the following reasons. Applicant's argument is that the extended stack instructions are part of a single instruction set of Feierbach. The examiner will rebut the appellant's arguments first with a definition of instruction set.

Instruction set is a list of all instructions that a processor can execute. The appellant claims to have a processor with two instruction sets, which might seem to go against this definition. However, a processor can have multiple cores of execution that execute different types of instructions, with each core executing a single instruction set. Each core executes a separate instruction set because each core was originally designed to execute a specific set of instructions. In addition, the instruction sets can overlap in functionality since each processor core is specifically designed to execute a

particular set of instructions. The claimed invention allows for such overlap since it doesn't declare no overlap can occur.

Feierbach disclosed two such processor cores by elements 202 and 204 of figure 2. These elements are two processor cores that are capable of executing two separate instruction sets since each processor core was specifically designed to execute a set of instructions. Feierbach disclose in column 5 lines 62-67 continued to column 6 lines 1-7 that stack instructions are executed as the first instruction set on processor core element 202 and that extended instructions are executed primarily as register-based instructions on processor core element 204. Thus, Feierbach, in view of the definition of instruction set, clearly disclosed two separate instruction sets in elements 202 and 204 of figure 2.

52. Regarding claims 8-11 rejected under 35 U.S.C. §102(b) as being anticipated by Feierbach et al. (U.S. 6,088,786):

Appellant makes similar arguments for claims 8-11 describing how Feierbach only disclosed a single instruction set for the processor core elements 202 and 204. The examiner directs the board to the section A rebuttal in paragraph 51 to show how Feierbach disclosed two separate instruction sets via processor core elements 202 and 204.

53. Regarding claims 12-15, 18-24, and 27 rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Patel et al. (U.S. 6,826,749), in view of Batten et al. (U.S. 6,256,725) in view of Hennessy et al. ("Computer Architecture: A Quantitative Approach"):

Appellant makes similar arguments for claims 12-15, 18-24, and 27 describing how Feierbach only disclosed a single instruction set for the processor core elements 202 and 204. The examiner directs the board to the section A rebuttal in paragraph 51 to show how Feierbach disclosed two separate instruction sets via processor core elements 202 and 204.

54. Regarding claims 30-35, 37, and 41 rejected under 35 U.S.C. §103(a) as being unpatentable over Feierbach et al. (U.S. 6,088,786), in view of Hendler et al. (U.S. 6,473,777):

Appellant makes similar arguments for claims 30-35, 37, and 41 describing how Feierbach only disclosed a single instruction set for the processor core elements 202 and 204. The examiner directs the board to the section A rebuttal in paragraph 51 to show how Feierbach disclosed two separate instruction sets via processor core elements 202 and 204.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/JAP/

/Jacob A. Petranek/

January 23, 2008

Conferees:

~~Eddie Chan~~

MATTHEW KIM
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Supervisory Patent Examiner

Technology Center 2100

Lynne Browne

Appeal Practice Specialist, TQAS

Technology Center 2100